Michael Potter

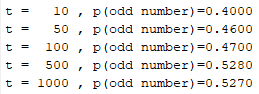
ENGR131A-80

UID 705644667

Final Project (**based on first Class Project upload**)

1.

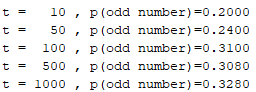
a.



b.

c. The experimental result approaches the theoretical result as the number of coin tosses increases (and they are approximately equal as t goes to infinity due to law of large numbers).

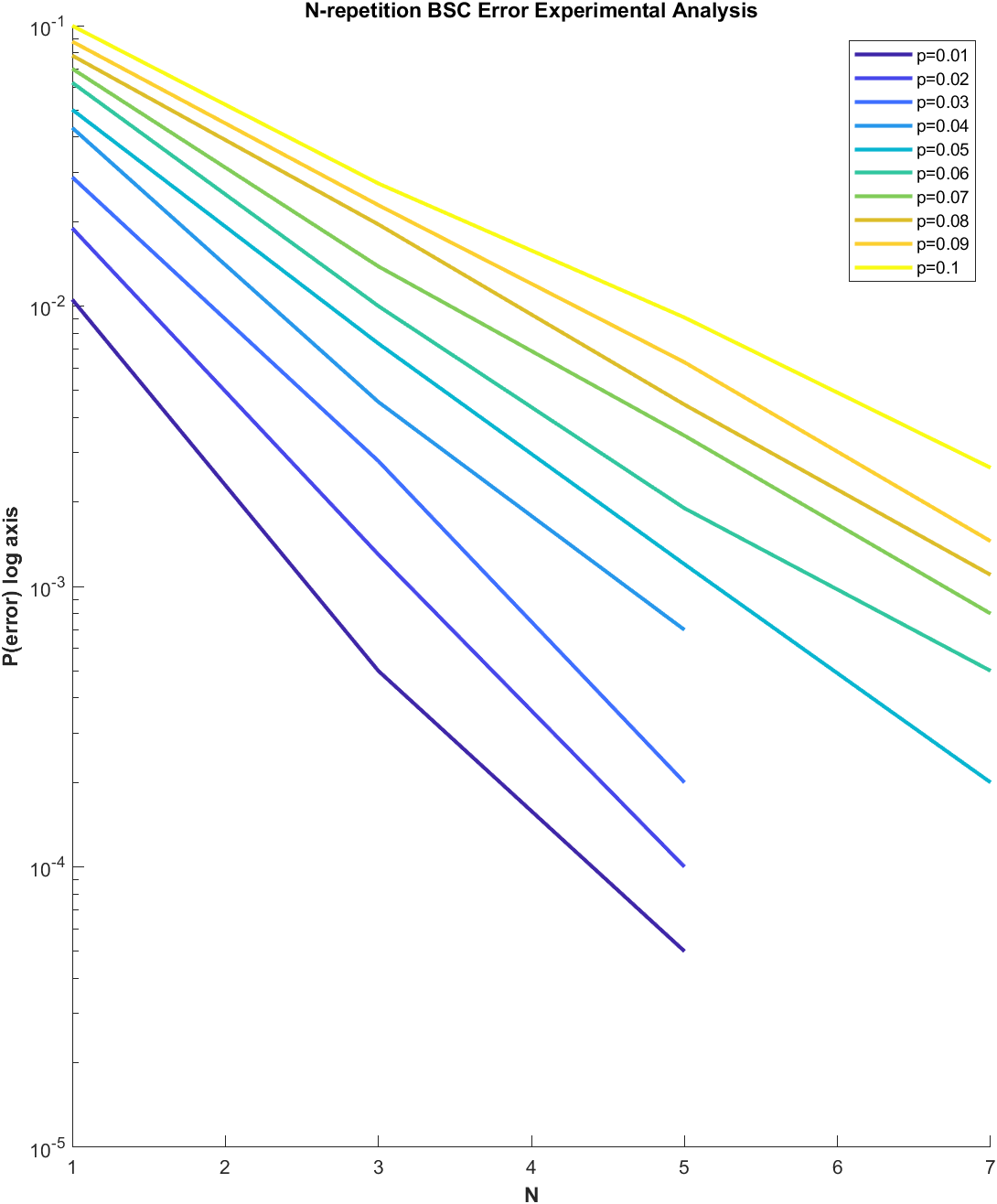
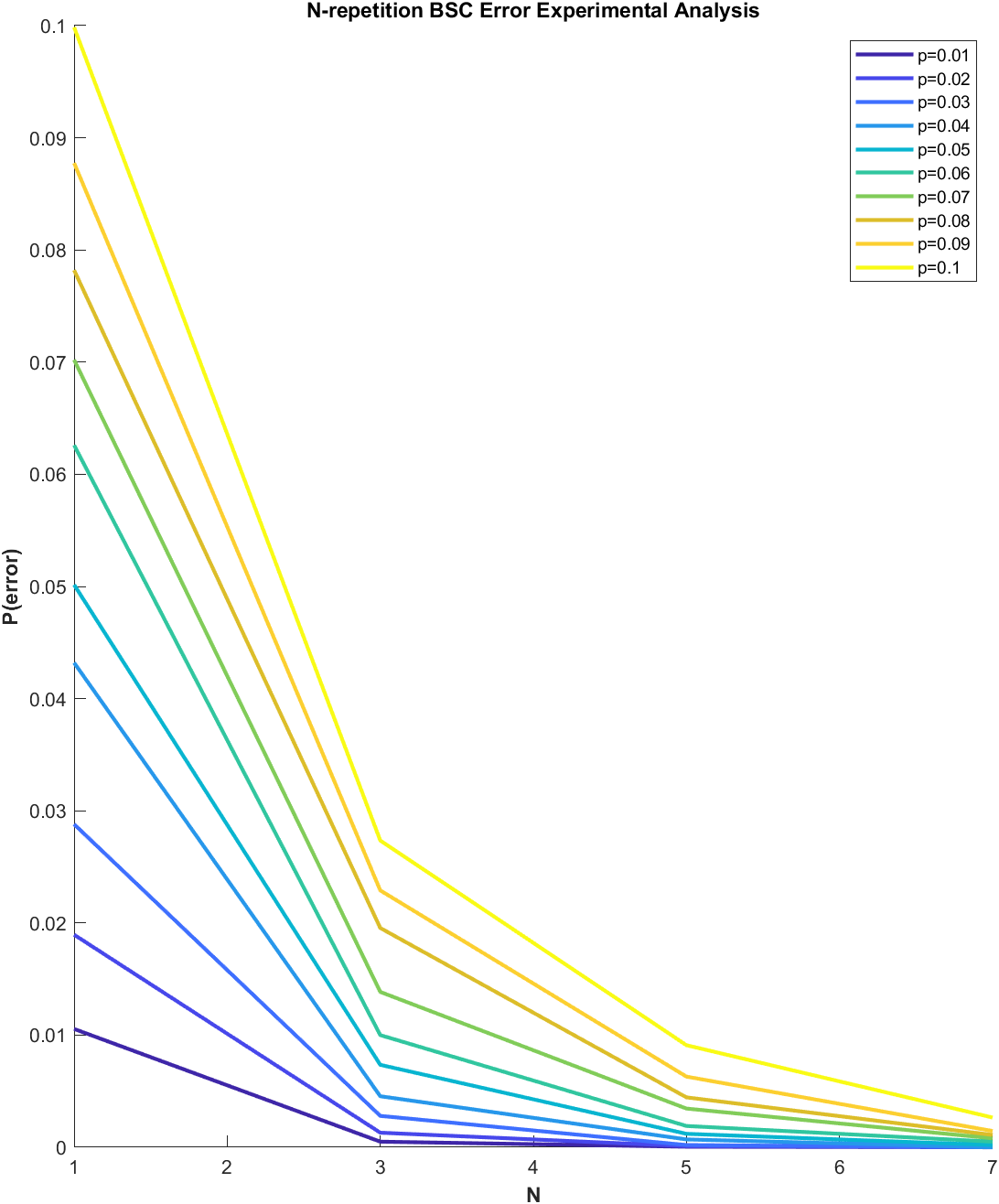
d.

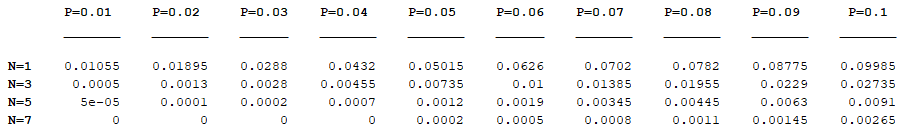


Similarly to part c, the experimental result approaches the theoretical result as the number of coin tosses increases (and they are approximately equal as t goes to infinity due to law of large numbers).

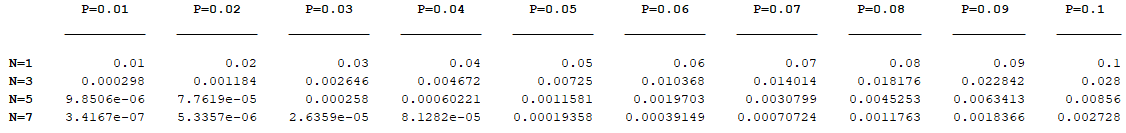
2.

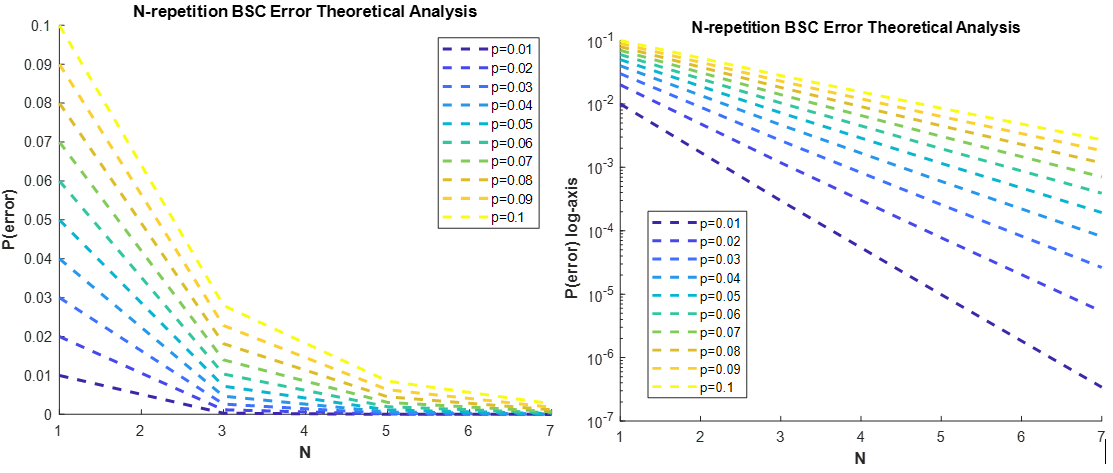
a.





b.





c. If the BSC channel has the probability of flipping a bit equal to 0.5, then the BSC channel is uninterpretable (a random channel), as the user cannot determine what bit was sent. To show this, follow the proof below:

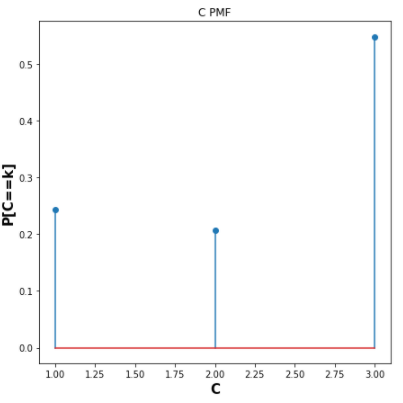
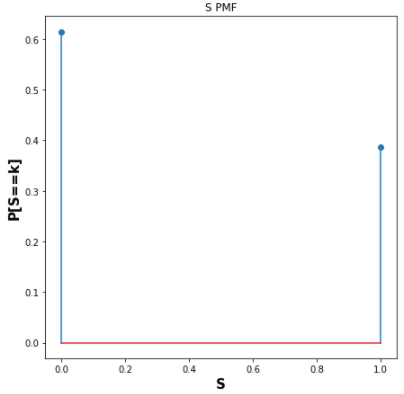
Then, by the same logic as the derivation above, p(R=1 | S=1) = p(R=0 | S=1)

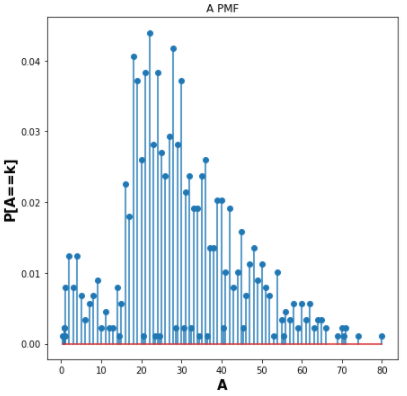
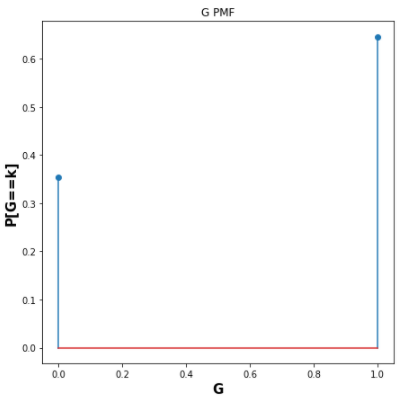
and the same logic holds for p(S=0 | R = 1) = p(S=1 | R=1).

If I knew that 0.5 < p < 1 then I would say that the receiver should swap the 1s with the 0s and vice versa to get an equivalent channel cross probability of (1-p) < ½ .

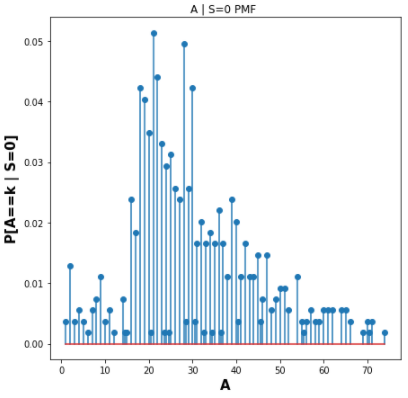
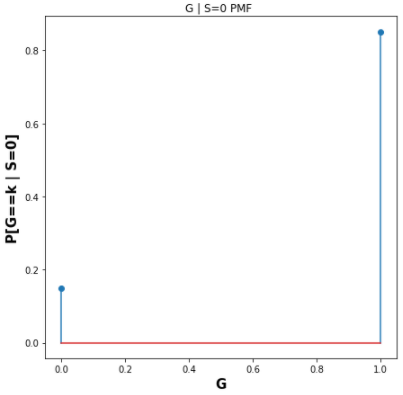
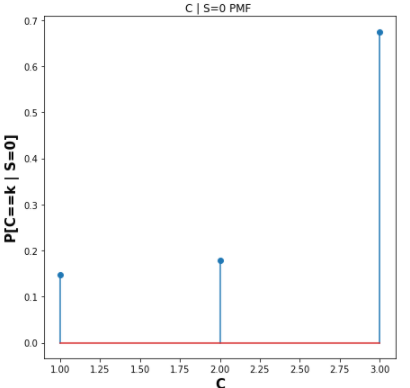
3.

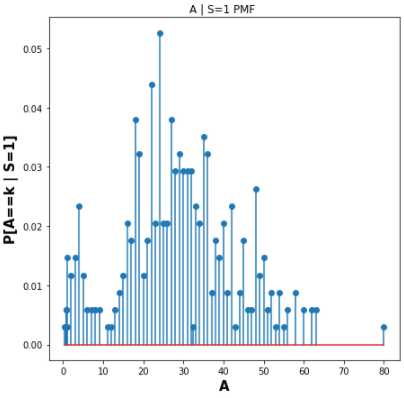
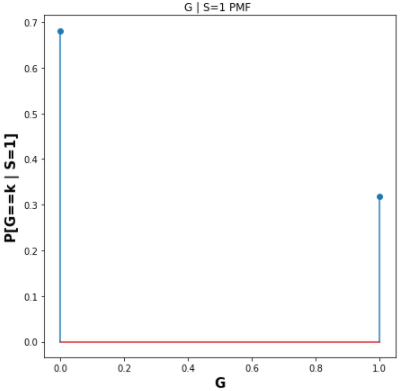
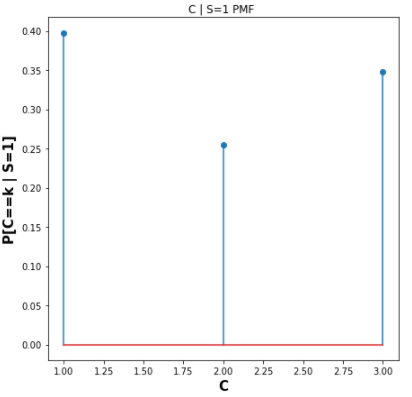
a.





b.





c.

**0.010625322687488965**

**0.08460553314142813**

d.

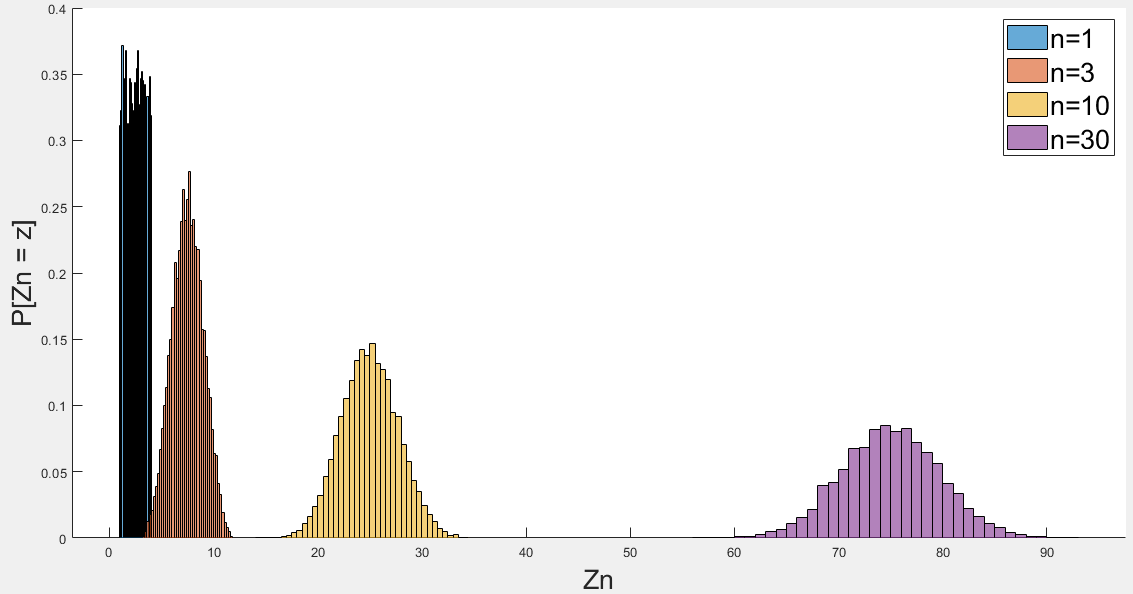
**0.11157436941003064**

**0.8884256305899694**

She will survive with probability of approximately 0.888 (most likely will survive…) .

4.

a.

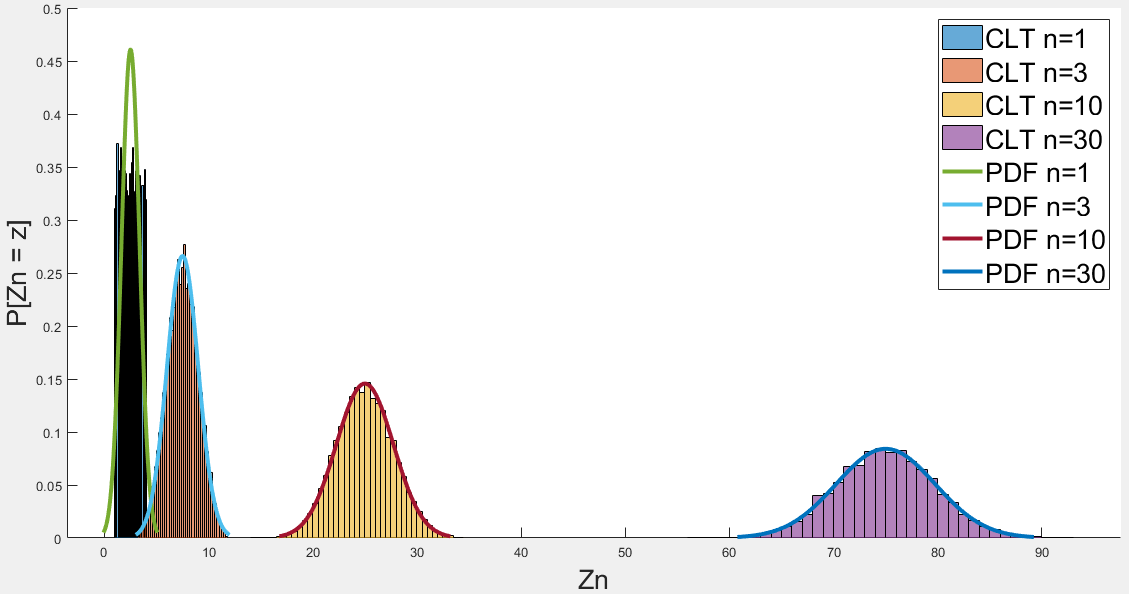


As n increases (and approaches infinity) the distribution of becomes a Gaussian distribution. This is evident by looking at the difference between n = 1, and n = 30. At n = 1, the distribution of appears to still be a uniform distribution like the iid , but when n = 30, the distribution of appears to be a Gaussian distribution. The tails become wider and wider as n increases, with the peak probability at the mean decreasing.

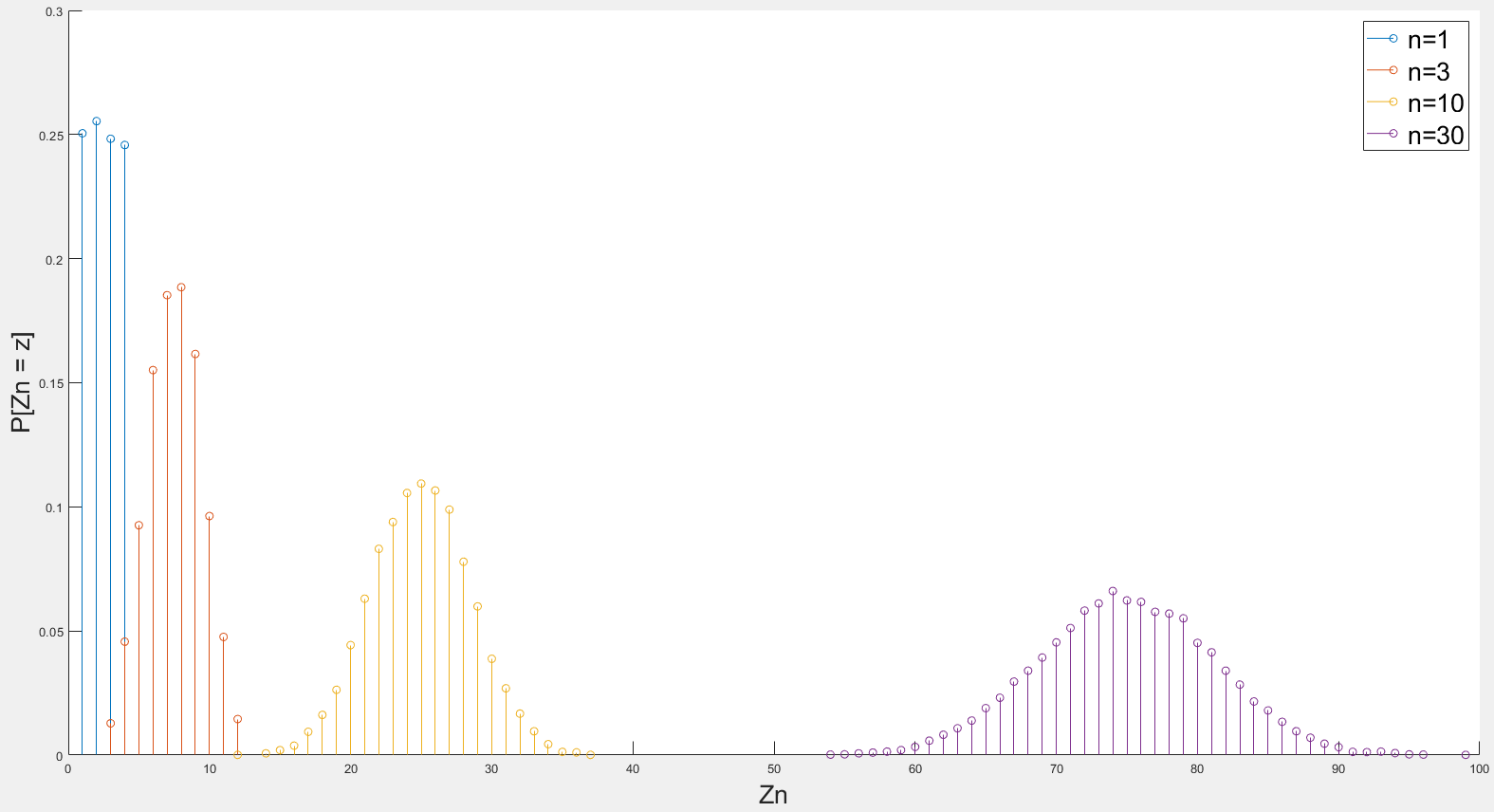
b.

|  |  |  |
| --- | --- | --- |
| **n** | **E[Zn]** | **Var(Zn)** |
| 1 | 2.5 | 0.75 |
| 3 | 7.5 | 2.25 |
| 10 | 25.0 | 7.50 |
| 30 | 75.0 | 22.5 |

c.

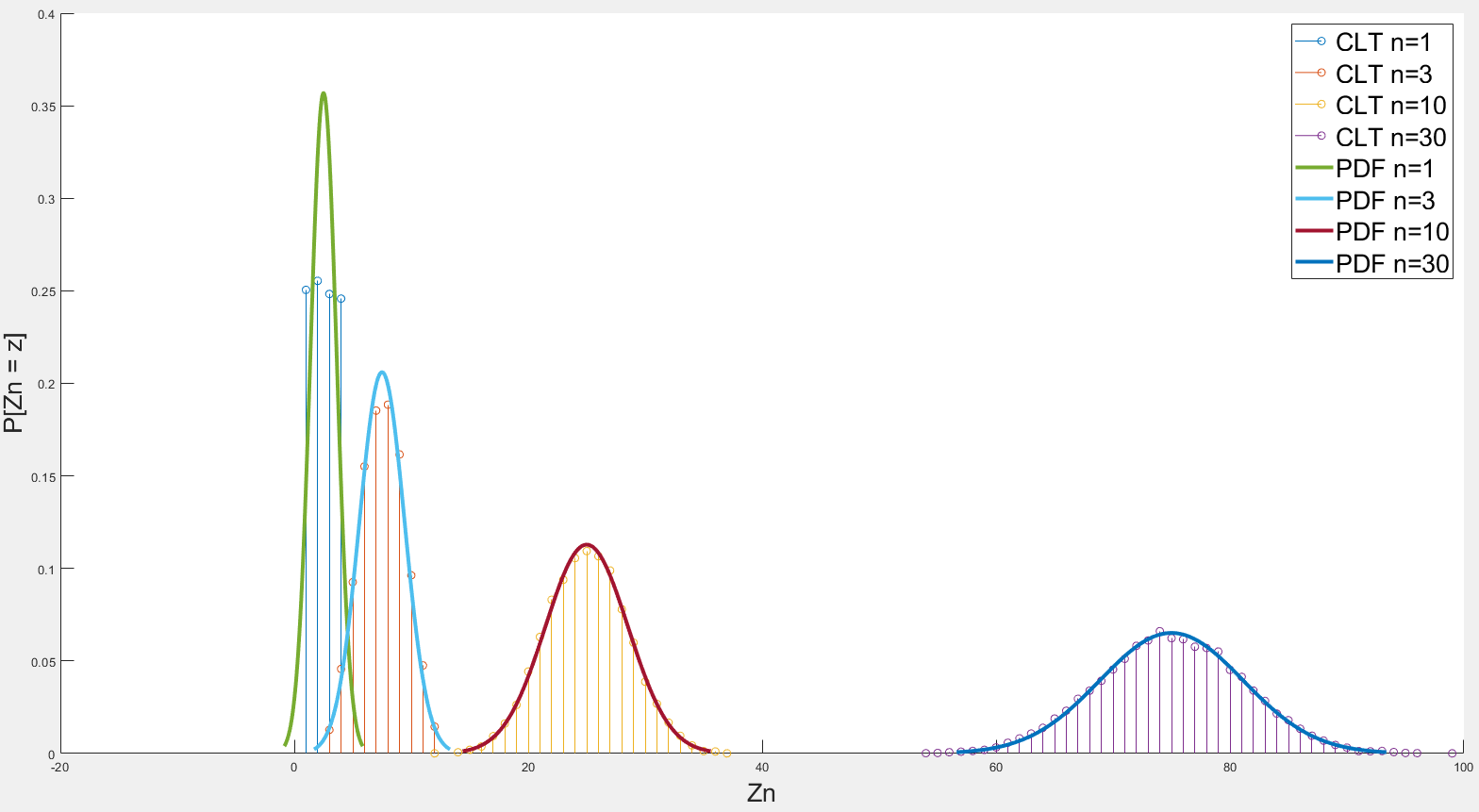


d.



As n increases (and approaches infinity) the distribution of becomes a Gaussian distribution. This is evident by looking at the difference between n = 1, and n = 30. At n = 1, the distribution of appears to still be a uniform distribution like the iid , but when n = 30, the distribution of appears to be a Gaussian distribution. The tails become wider and wider as n increases, with the peak decreasing. A difference between part (a) is that Zn is discrete random variable.

|  |  |  |
| --- | --- | --- |
| **n** | **E[Zn]** | **Var(Zn)** |
| 1 | 2.5 | 1.25 |
| 3 | 7.5 | 3.75 |
| 10 | 25.0 | 12.5 |
| 30 | 75.0 | 37.5 |



Appendix (Code)

**1.**

%% 1a

close all; clear all; clc;

tosses = [10,50,100,500,1000];

for t = tosses

p\_odd = mean(mod(randi(4,1,t),2));

fprintf("t =%5d , p(odd number)=%.4f\n",t,p\_odd)

end

%% 1d

close all; clear all; clc;

tosses = [10,50,100,500,1000];

s = RandStream('mlfg6331\_64');

for t = tosses

p\_odd = mean(mod(datasample(s,1:4,t,'Weights',[1/6,2/6,1/6,2/6]),2));

fprintf("t =%5d , p(odd number)=%.4f\n",t,p\_odd)

end

**2.**

close all; clear all; clc;

%%

p = 0.01:0.01:.1;

N = 1:2:7;

for i = 1:length(N)

for j = 1:length(p)

[bits\_sent,bits\_encoded] = transmitted(p(j),N(i));

p\_error(i,j) = mean(decoder(bits\_sent,N(i)) ~= bits\_encoded);

theoretical\_error(i,j) = p\_error\_theor(p(j),N(i));

end

end

subplot(1,2,1)

hold on

cmap = colormap(parula(length(p)));

for i = 1:length(p)

plot(N,p\_error(:,i),'Color',cmap(i,:),'LineWidth',2)

end

legend("p="+string(p))

title("N-repetition BSC Error Experimental Analysis")

xlabel("N",'FontWeight','bold')

ylabel("P(error)",'FontWeight','bold')

figure(1)

subplot(1,2,2)

hold on

cmap = colormap(parula(length(p)));

for i = 1:length(p)

plot(N,p\_error(:,i),'Color',cmap(i,:),'LineWidth',2)

end

set(gca,'yscale','log')

legend("p="+string(p))

title("N-repetition BSC Error Experimental Analysis")

xlabel("N",'FontWeight','bold')

ylabel("P(error) log axis",'FontWeight','bold')

array2table(p\_error,'RowNames',"N="+string(N),'VariableNames',"P="+string(p))

%%

figure(2)

subplot(1,2,2)

hold on

for i = 1:length(p)

plot(N,theoretical\_error(:,i),'Color',cmap(i,:),'LineStyle','--','LineWidth',2)

end

legend("p="+string(p))

set(gca,'yscale','log')

legend("p="+string(p))

title("N-repetition BSC Error Theoretical Analysis")

xlabel("N",'FontWeight','bold')

ylabel("P(error) log axis",'FontWeight','bold')

subplot(1,2,1)

hold on

for i = 1:length(p)

plot(N,theoretical\_error(:,i),'Color',cmap(i,:),'LineStyle','--','LineWidth',2)

end

legend("p="+string(p))

legend("p="+string(p))

title("N-repetition BSC Error Theoretical Analysis")

xlabel("N",'FontWeight','bold')

ylabel("P(error)",'FontWeight','bold')

array2table(theoretical\_error,'RowNames',"N="+string(N),'VariableNames',"P="+string(p))

%%

function [bits\_sent,bits\_encoded] = transmitted(p,N)

bits\_encoded = rand(20000,1) < .5; % <.5 is 1, >= .5 is 0 transmitted

bits\_sent = rand(20000,N);

bits\_sent = (bits\_sent < p) .\* (1-bits\_encoded) + (1-(bits\_sent < p)) .\* bits\_encoded;

end

function bits\_received = decoder(transmission,N)

bits\_received = sum(transmission,2)>N/2;

end

function theoretical\_error = p\_error\_theor(p,N)

n = ceil(N/2);

theoretical\_error = 0;

for i = n:N

theoretical\_error = theoretical\_error + nchoosek(N,i)\*p^i\*(1-p)^(N-i);

end

end

**3. (in python)**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import os

titanic = pd.read\_csv("titanic.csv").rename(columns={'Survived':'S','Pclass':'C','Sex':'G','Age':'A'})

titanic.head()

titanic.describe()

titanic.isna().sum()

# a

N = titanic.shape[0]

PMF = {}

for col in titanic.columns:

plt.figure(figsize=(7,7))

ps = titanic[col].value\_counts()/N

plt.stem(ps.index,ps)

plt.title("{} PMF".format(col))

plt.xlabel(col,fontsize=15,weight='bold')

plt.ylabel("P[{}==k]".format(col),fontsize=15,weight='bold')

PMF[col] = ps

# b

survived = titanic.pop("S")

survived.unique()

PMF\_Conditioned = {}

for survival in survived.unique():

for col in titanic.columns:

plt.figure(figsize=(7,7))

N = (survived==survival).sum()

ps = (titanic[col][survived==survival]).value\_counts() / N

plt.stem(ps.index,ps)

plt.title("{} | S={} PMF ".format(col,survival))

plt.xlabel(col,fontsize=15,weight='bold')

plt.ylabel("P[{}==k | S={}]".format(col,survival),fontsize=15,weight='bold')

PMF\_Conditioned["{}|S={}".format(col,survival)] = ps

# c

p0 = PMF['S'][0]\*((PMF\_Conditioned["A|S=0"][PMF\_Conditioned["A|S=0"].index<=40]).sum())\*(PMF\_Conditioned["C|S=0"][1])\*(PMF\_Conditioned["G|S=0"][0])

p1 = PMF['S'][1]\*((PMF\_Conditioned["A|S=1"][PMF\_Conditioned["A|S=1"].index<=40]).sum())\*(PMF\_Conditioned["C|S=1"][1])\*(PMF\_Conditioned["G|S=1"][0])

# d (she will survive)

p0 / (p0 + p1)

p1 / (p0 + p1)

**4.**

%% a

close all; clc; clear all;

samples = [1,3,10,30];

i = 1;

for n = samples

xs = rand(10000,n)\*3 + 1;

Z = sum(xs,2);

fprintf("Z @ n =%3d = %4.5f ; =%4.5f\n",n,mean(Z),var(Z))

z(i,:) = Z;

i = i + 1;

end

figure(1)

hold on

for i = 1:length(samples)

histogram(z(i,:),'Normalization','pdf')

end

legend("n="+string(samples),'FontSize',20)

fprintf("\n\n")

ylabel("P[Zn = z]",'FontSize',20)

xlabel("Zn",'FontSize',20)

%% b

clearvars -except samples

i = 1;

for n = samples

mu = (4+1)/2;

var = (4-1)^2 / 12;

z\_mu(i) = (mu\*n);

z\_var(i) = (var\*n);

fprintf("Z\_mu @ n = %3d = %4.5f\n",n,z\_mu(i))

fprintf("Z\_var @ n = %3d = %4.5f\n\n",n,z\_var(i))

i = i + 1;

end

%% c

figure(1)

hold on

for i = 1:length(samples)

x = linspace(z\_mu(i) - sqrt(z\_var(i))\*3,z\_mu(i) + sqrt(z\_var(i))\*3,1000);

y = normpdf(x,z\_mu(i),sqrt(z\_var(i)));

plot(x,y,'linewidth',3)

end

legend('CLT n=1','CLT n=3','CLT n=10','CLT n=30','PDF n=1','PDF n=3','PDF n=10','PDF n=30')

%% a

close all; clc; clear all;

samples = [1,3,10,30];

i = 1;

for n = samples

xs = randi([1,4],10000,n);

Z = sum(xs,2);

fprintf("Z @ n =%3d = %4.5f ; =%4.5f\n",n,mean(Z),var(Z))

z(i,:) = Z;

i = i + 1;

end

%%

figure(1)

hold on

for i = 1:length(samples)

[C,ia,ic] = unique(z(i,:));

a\_counts = accumarray(ic,1);

value\_counts = [C', a\_counts];

stem(C',a\_counts/sum(a\_counts));

end

legend("n="+string(samples),'FontSize',20)

fprintf("\n\n")

ylabel("P[Zn = z]",'FontSize',20)

xlabel("Zn",'FontSize',20)

%% b

clearvars -except samples

i = 1;

for n = samples

mu = (4+1)/2;

var = ((4-1+1)^2 - 1) / 12;

z\_mu(i) = (mu\*n);

z\_var(i) = (var\*n);

fprintf("Z\_mu @ n = %3d = %4.5f\n",n,z\_mu(i))

fprintf("Z\_var @ n = %3d = %4.5f\n\n",n,z\_var(i))

i = i + 1;

end

%% c

figure(1)

hold on

for i = 1:length(samples)

x = linspace(z\_mu(i) - sqrt(z\_var(i))\*3,z\_mu(i) + sqrt(z\_var(i))\*3,1000);

y = normpdf(x,z\_mu(i),sqrt(z\_var(i)));

plot(x,y,'linewidth',3)

end

legend('CLT n=1','CLT n=3','CLT n=10','CLT n=30','PDF n=1','PDF n=3','PDF n=10','PDF n=30')